Oracle® Retail Demand Forecasting

Administration Guide

Release 13.1.1

September 2009

ORACLE

Oracle® Demand Forecasting Administration Guide, Release 13.1.1

Primary Author: Melissa Artley

Licensing Note: This media pack includes a Restricted Use license for Oracle Retail Predictive Application Server (RPAS) - Enterprise Engine to support Oracle® Retail **Demand Forecasting** only.

Value-Added Reseller (VAR) Language

Oracle Retail VAR Applications

The following restrictions and provisions only apply to the programs referred to in this section and licensed to you. You acknowledge that the programs may contain third party software (VAR applications) licensed to Oracle. Depending upon your product and its version number, the VAR applications may include:

(i) the software component known as **ACUMATE** developed and licensed by Lucent Technologies Inc. of Murray Hill, New Jersey, to Oracle and imbedded in the Oracle Retail Predictive Application Server – Enterprise Engine, Oracle Retail Category Management, Oracle Retail Item Planning, Oracle Retail Merchandise Financial Planning, Oracle Retail Advanced Inventory Planning, Oracle Retail Demand Forecasting, Oracle Retail Regular Price Optimization, Oracle Retail Size Profile Optimization, Oracle Retail Replenishment Optimization applications.

(ii) the **MicroStrategy** Components developed and licensed by MicroStrategy Services Corporation (MicroStrategy) of McLean, Virginia to Oracle and imbedded in the MicroStrategy for Oracle Retail Data Warehouse and MicroStrategy for Oracle Retail Planning & Optimization applications.

(iii) the **SeeBeyond** component developed and licensed by Sun MicroSystems, Inc. (Sun) of Santa Clara, California, to Oracle and imbedded in the Oracle Retail Integration Bus application.

(iv) the **Wavelink** component developed and licensed by Wavelink Corporation (Wavelink) of Kirkland, Washington, to Oracle and imbedded in Oracle Retail Mobile Store Inventory Management.

(v) the software component known as **Crystal Enterprise Professional and/or Crystal Reports Professional** licensed by SAP and imbedded in Oracle Retail Store Inventory Management.

(vi) the software component known as **Access Via**™ licensed by Access Via of Seattle, Washington, and imbedded in Oracle Retail Signs and Oracle Retail Labels and Tags.

(vii) the software component known as **Adobe Flex**™ licensed by Adobe Systems Incorporated of San Jose, California, and imbedded in Oracle Retail Promotion Planning & Optimization application.

(viii) the software component known as **Style Report**™ developed and licensed by InetSoft Technology Corp. of Piscataway, New Jersey, to Oracle and imbedded in the Oracle Retail Value Chain Collaboration application.

(ix) the software component known as **DataBeacon**™ developed and licensed by Cognos Incorporated of Ottawa, Ontario, Canada, to Oracle and imbedded in the Oracle Retail Value Chain Collaboration application.

You acknowledge and confirm that Oracle grants you use of only the object code of the VAR Applications. Oracle will not deliver source code to the VAR Applications to you. Notwithstanding any other term or condition of the agreement and this ordering document, you shall not cause or permit alteration of any VAR Applications. For purposes of this section, "alteration" refers to all alterations, translations, upgrades, enhancements, customizations or modifications of all or any portion of the VAR Applications including all reconfigurations, reassembly or reverse assembly, re-engineering or reverse engineering and recompilations or reverse compilations of the VAR Applications or any derivatives of the VAR Applications. You acknowledge that it shall be a breach of the agreement to utilize the relationship, and/or confidential information of the VAR Applications for purposes of competitive discovery.

The VAR Applications contain trade secrets of Oracle and Oracle's licensors and Customer shall not attempt, cause, or permit the alteration, decompilation, reverse engineering, disassembly or other reduction of the VAR Applications to a human perceivable form. Oracle reserves the right to replace, with functional equivalent software, any of the VAR Applications in future releases of the applicable program.

Contents

Preface

Oracle Retail Administration Guides are designed so that you can view and understand the application's 'behind-the-scenes' processing, including such information as the following:

- Key system administration configuration settings
- Technical architecture
- Functional integration dataflow across the enterprise
- Batch processing

Audience

Anyone who has an interest in better understanding the inner workings of the RDF system can find valuable information in this guide. There are three audiences in general for whom this guide is written:

- System analysts and system operation personnel:
  - who are looking for information about RDF processes internally or in relation to the systems across the enterprise.
  - who operate RDF on a regular basis.
- Integrators and implementation staff who have the overall responsibility for implementing RDF into their enterprise.
- Business analysts who are looking for information about processes and interfaces to validate the support for business scenarios within RDF and other systems across the enterprise.

Related Documents

For more information, see the following documents in the Oracle Retail Demand Forecasting Release 13.1.1 documentation set:

- *Oracle Retail Demand Forecasting Release Notes*
- *Oracle Retail Demand Forecasting Installation Guide*
- *Oracle Retail Demand Forecasting Configuration Guide*
- *Oracle Retail Demand Forecasting User Guide*
- Oracle Retail Predictive Application Server documentation

Customer Support

To contact Oracle Customer Support, access My Oracle Support at the following URL:

https://metalink.oracle.com

When contacting Customer Support, please provide the following:

- Product version and program/module name
- Functional and technical description of the problem (include business impact)
- Detailed step-by-step instructions to re-create
- Exact error message received
- Screen shots of each step you take

Review Patch Documentation

If you are installing the application for the first time, you install either a base release (for example, 13.0) or a later patch release (for example, 13.0.2). If you are installing a software version other than the base release, be sure to read the documentation for each patch release (since the base release) before you begin installation. Patch documentation can contain critical information related to the base release and code changes that have been made since the base release.

Oracle Retail Documentation on the Oracle Technology Network

In addition to being packaged with each product release (on the base or patch level), all Oracle Retail documentation is available on the following Web site (with the exception of the Data Model which is only available with the release packaged code):

http://www.oracle.com/technology/documentation/oracle_retail.html

Documentation should be available on this Web site within a month after a product release. Note that documentation is always available with the packaged code on the release date.

Conventions

Navigate: This is a navigate statement. It tells you how to get to the start of the procedure and ends with a screen shot of the starting point and the statement "the Window Name window opens."

> **Note:** This is a note. It is used to call out information that is important, but not necessarily part of the procedure.

```
This is a code sample
   It is used to display examples of code
```

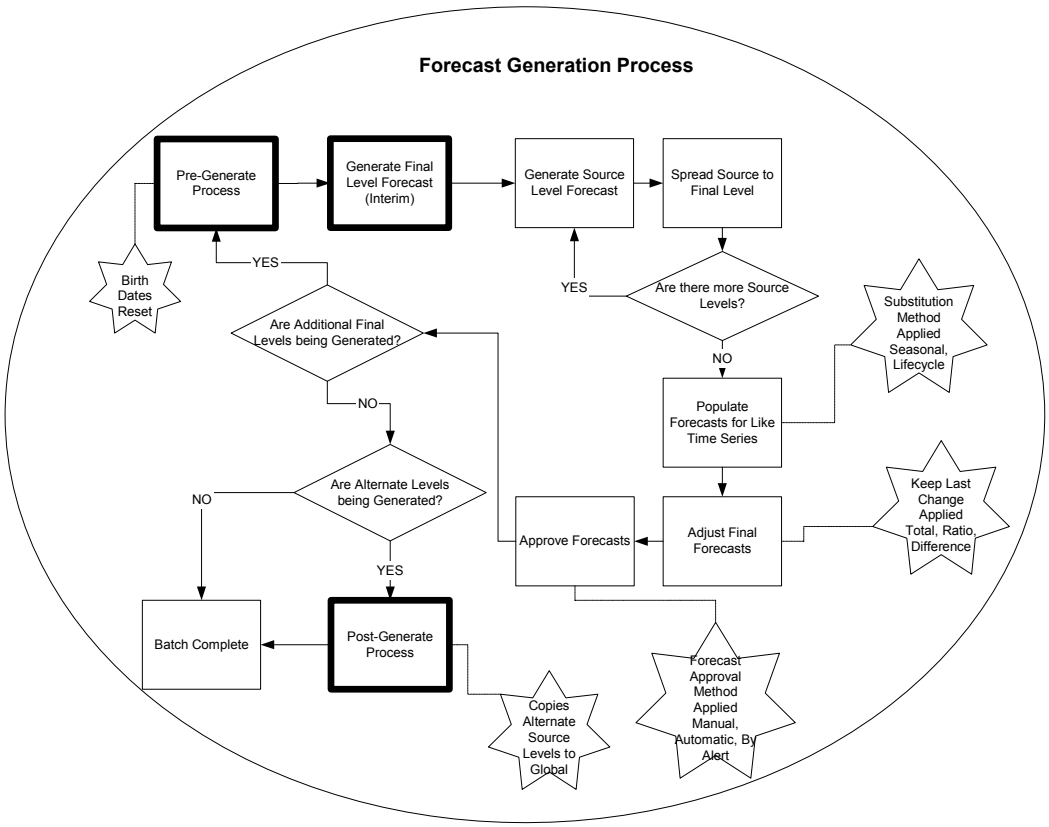
A hyperlink appears like this.

1

Batch Processes

Overview

The diagram below provides a high-level overview of the batch forecast process.



Overview of Batch Forecast Process

Batch Processes

PreGenerateForecast

Used in a Global Domain or Simple Domain environment, “PreGenerateForecast” is an RDF executable that registers all measures with a birth date prior to forecast generation using “generate.” The first time “PreGenerateForecast” is run for a level, it registers the appropriate token measures for that level. If a Global Domain environment is implemented, “PreGenerateForecast” may be run against the Master or a Local domain. At either level, the necessary measures to produce the batch forecast will be registered across all domains.

“PreGenerateForecast” requires an input file in the form of an XML. The XML is configured with the following values:

- **FinalLevel:** The Final Level Number that will be used to generate the forecast.
- **OutputFile:** The name of the resulting file located at the root of the domain after PreGenerateForecast is run. The OutputFile includes the values set for FinalLevel and Override in addition to the birth date. This date is the Forecast Generation Date, and it is passed to the domains when ‘generate’ is run.

  The date is produced in the following format: yyyymmddHhhMmm (Example: 20050327H13M36). When this birth date is selected in the Forecast Approval wizard, it will be viewed as: (03/27/2005 13:36).
- **Override:** A true or false value. When “generate” is passed a true value, the Next Run Date is ignored, and the batch forecast uses today’s date as the Next Run Date; and the batch is executed. When ‘generate’ is passed a false value, the batch forecast will run if the Next Run Date is the same as today’s date.

> **Note:** When the **Run Batch** template is used to generate the batch forecast, PreGenerateForecast is run automatically. If a Global Domain environment is implemented, forecasts produced across Local domains using “Run Batch” cannot be aggregated in the Master domain because they do not share the same Forecast Generation Date.

Usage

`PreGenerateForecast -InputFile filename`

`InputFile` is required.

The input file should be an XML file that looks like this:

```
<Parameters>
   <Parameter>
      <Key>FinalLevel</Key>
      <Value>1</Value>
 </Parameter>

<Parameter>
   <Key>OutputFile</Key>
   <Value>MyOutput.xml</Value>
 </Parameter>
 <Parameter>
   <Key>Override</Key>
   <Value>true</Value>
   </Parameter>
</Parameters>
```

`FinalLevel` and `OutputFile` are required parameters of the XML file.

`Override` is an optional parameter of the XML file (default is false).

Other parameters may be included in the input XML file. They will be passed through to the output XML file.

Return codes:

- 0 – Success (either ran pre-generate or did not need to run)
- 1 – Bad input
- 2 – Failure

To set the logger verbosity level, use `-loglevel` with values of: `all`, `profile`, `debug`, `information`, `warning`, `error`, or `none`. To disable timestamp header use `-noheader`.

generate

"generate" is an RDF executable used to produce the batch forecast. This executable requires as an input, the OutputFile resulting from "PreGenerateForecast."

This binary runs RDF's batch process. Generate can take two optional inputs: level and override.

Usage:

`generate -InputFile Filename`

The following parameters setting are included in the input file: `birth`, `startdate`, `finallevel`, and `override`.

The override input must be true or false. The defaulted value is false if this option is not included in the input file. When override is false, generate will only start the batch process if current time is later than the next run date in the domain. When the override is true, generate will start the batch forecast regardless of the next run date.

The generate binary invokes code in the BatchForecast library to execute the batch process.

`finalLevel` and `birth` are required parameters of the XML file. `override` (false) and `StartDate` (Default Forecast Start Date) are optional parameters of the XML file (defaults in parentheses).

Return codes:

- 0 – Success (either ran generate or did not need to run)
- 1 – Bad input
- 2 – Failure

To set the logger verbosity level, use `-loglevel` with values of: `all`, `profile`, `debug`, `information`, `warning`, `error`, or `none`.

To disable timestamp header use `-noheader`.

The input file should be an XML file that looks similar to the following:

```
<Parameters>
   <Parameter>
       <Key>Birth</Key>
       <Value>20041027H11M52</Value>
 </Parameter>
 <Parameter>
      <Key>StartDate</Key>
      <Value>DAY20041027</Value>
 </Parameter>
 <Parameter>
       <Key>FinalLevel</Key>
       <Value>1</Value>
 </Parameter>
 <Parameter>
      <Key>Override</Key>
      <Value>true</Value>
   </Parameter>
</Parameters>
```

RDFvalidate

`RDFvalidate` automatically executes during the domain install, and it can also be run at any time against a Master or one subdomain. If run against the Master Domain, it checks the master and all subdomains. If run against a subdomain, it checks the Master and only the subdomain (not all other subdomains). This function verifies that:

- If there is a partition dimension, it must be along the product hierarchy.
- Domains are cleanly partitioned, this means that for the partition dimension, there exists only one position in each local domain, whether partitioning along the main or an alternate (or branch) product hierarchy.
- All data, measures, and levels are defined properly based on the partition dimension
- Causal parameters are properly defined based on final, source, and causal levels

Usage

```
rdfvalidate -d pathToDomain
```

To get this usage text, use `-?`, `-help`, or `-usage`. To get the version of this utility, use –version. To set the logger verbosity level, use `-loglevel` with values of: `all`, `profile`, `debug`, `information`, `warning`, `error`, or `none`. To disable timestamp header use `-noheader`.

RDF Validation

The list below displays the validation performed internally by the `RDFvalidate` utility.

1. Hierarchies and Dimensions:
   a. Verify “day” dimension exists on calendar hierarchy.
   b. If there is a partition dimension, it must be along the product hierarchy.
2. For final levels:
   a. Intersection (fintxlxb)
      - Cannot be blank
      - Must be at or below all source level intersections
      - Must be at or below the partition dimension on the partition branch
   b. Seasonal profile (seasprofxlxb) can be either:
      - Blank
      - Measure name (only one)
        - Must be valid measure
        - Should be of type real
        - Measure intersection must be equal to the level intersection
   c. Source data (datasrcxlxb) must be a measure name (only one)
      - Must be a valid measure
      - Should be of type real
      - Measure intersection must be at or below the final level intersection

   d. Plan data (r fplanxlxb) must be either:
      - Blank
      - Measure name (only one)
         - Must be valid measure
         - Should be of type real
         - Measure intersection must be equal to the final level intersection
3. For source levels:
   a. Intersection (fintxlxb)
      - Cannot be blank
      - Must be at or above final level intersection
      - Must contain a dimension from the partition hierarchy
      - Must be either:
         - At or below the partition dimension on the partition branch.
         - On a branch of the partition hierarchy.
           If on a branch of the partition hierarchy, also check if domains are "cleanly" partitioned (executable only). This means for the branched dimension on the partition hierarchy, each position for that dimension can exist in only one sub-domain.
   b. Seasonal profile (seasprofxlxb) can be either:
      - Blank
      - Measure name (only one)
         - Must be valid measure
         - Should be of type real
         - Measure intersection must be equal to the level intersection
   c. Spreading profiles (sprdprofxlxb)
      - Can only be blank if source level intersection equals final level intersection
      - Must be comma-separated list of curve levels and measure names (can be mixed)
         - If curve level, must be a valid curve level (final profile)
         - If measure:
            - Must be a valid measure
            - Should be of type real
            - Measure intersection must be at or above final level

Executable Only

1. Make sure there is only one partition dimension per subdomain.
2. Final and source levels:
   a. Causal Aggregation Profile (aggxlxb) values should be either:
      - Blank
      - Measure name (one only)
         - Should be a valid measure
         - Should be of type real
         - The intersection of the measure must be at or above final level.

b. Causal Calculation Intersection (calcintxlxb) values should be either:
- Blank
- Intersection
  - Must be valid intersection
  - Must contain the calendar dimension
  - Must be at or above level intersection.

c. Causal Data Source (calcdtsrcxlxb) values should be either:
- Blank
- Measure name (one only)
  - Should be a valid measure
  - Should be of type real
  - The intersection of the measure must be at or above level intersection.

d. Causal Higher Intersection (cslhint) values should be either:
- Blank
- Intersection
  - Must be valid intersection
  - Must not contain the calendar dimension
  - Must contain a dimension from the partition hierarchy.
  - Must be at or above level intersection.
  - Must be either:
    - At or below the partition dimension on the partition branch.
    - On a branch of the partition hierarchy.

Note: If on a branch of the partition hierarchy, also check if domains are "cleanly" partitioned (executable only). This means that for the branched dimension on the partition hierarchy, each position for that dimension can exist in only one sub-domain.

e. Causal Spread Profile (spreadxlxb) values should be either:
- Blank
- Measure name (one only)
  - Should be a valid measure
  - Should be of type real
  - The intersection of the measure must be at or above final level.

f. Deseasonalized Demand Array (ddemandxlxb) values should be either:
- Blank
- Measure name (one only)
  - Should be a valid measure
  - Should be of type real
  - The intersection of the measure must be the level intersection less the calendar dimension

3. Final Levels only
   a. Default History Start Date (defhstdt) values should be either:
      – Blank
      – A date within the calendar
   b. Forecast Start Date (dfxlxb) values should be either:
      – Blank
      – A date within the calendar

Promote Validation

Plug-in and Executable

1. Hierarchies and Dimensions:

   Check whether or not "PTYP", "FLVL", and "PROM" exist in Data Hierarchy. If not, create them.
2. Promotion Names:

   Check if promotion names have 1-4 characters.
3. Causal levels must be at or below the partition dimension on the partition branch.

UpdateFnhbiRdf

`UpdateFnhbiRdf` is required after `Generate` is executed if an alternate hierarchy dimension from the Product hierarchy is used as a dimension in a forecast level. It performs the following functionality:

- Checks that certain measures are cleanly partitioned
- Copies corresponding cells (based on the partition) from each sub-domain to the master domain
- Runs automatically with the 'Run Batch' wizard
- After ensuring that the FNHBI (Forced non-Higher Based Intersections) measures are cleanly partitioned, UpdateFnhbiRdf copies corresponding cells (based on the partition dimension) from each sub-domain into the master domain

Usage

```
UpdateFnhbiRdf  -d pathToDomain -InputFile filename
```

To get this usage text, use `-?`, `-help`, or `-usage`. To get the version of this utility, use `-version`. To set the logger verbosity level, use `-loglevel` with values of: `all`, `profile`, `debug`, `information`, `warning`, `error`, or `none`. To disable timestamp header, use `-noheader`.

The `InputFile` format expected is as printed by the usage information. The timestamp or the birth key will have to be the same as the one output by pregenerateForecast, that is used by generate.

2

AutoSource

The AutoSource binary may be used to determine the optimal source level for a product/location. For the final level specified, AutoSource produces a forecast using each source level. The source level that produces the best MAE (Mean Absolute Error) for a time series is selected as the Optimal Source Level. The AutoSource results may be accessed by the user through the Forecast Maintenance workbook. If the Optimal Source Level is to be used for a product/location, the "Use Optimal Source" parameter should be set to true.

The AutoSource binary invokes code in the BatchForecast library to execute the batch process. AutoSource can take four inputs: mode (required), finallevels (required), today, and timelimit (required).

- Provides a starting Source Level recommendation for new forecasting customers. The recommended Source Level can be applied to the Final Level, which would allow the user to be focused on other tuning activities.
- Helpful for existing customers that are starting to forecast new businesses. AutoSource can be included as an activity in the customer's forecasting roll-out process.
- Useful for merchandise groups that have shifting demand patterns due to business or market changes such as pricing and marketing strategy changes, or product realignment.

AutoSource uses the forecast horizon to compute the MAE (Mean Absolute Error). If the forecast horizon is changed from the default of 13 weeks, AutoSource will start forecasting that number of weeks back. For example, if you have a forecast horizon of 52 weeks, AutoSource will start its analysis 53 weeks before today. This approach can disallow Winters and Seasonal models if sufficient calendar is not available. If the forecast horizon is 52 weeks, you should have at least 3.5 years of history for AutoSource to be able to perform all of its analysis.

Unlike Generate, there is no interim forecast calculation in AutoSource. Instead, AutoSource attempts to generate an AutoES result at the final level, then uses that result to perform the source level spreads. Note that if the time series data is dense enough at the final level, the spread will not be based on a linear contribution to the source. It will not maintain the source shape, and it will make recommendations based on such spreads.

AutoSource makes an initial recommendation to all the product/location combinations with sufficient data to perform analysis. Subsequent MAE calculation and comparison only occurs to these product/location combinations. The product/location combination without sufficient data (total sales = 0 during history region or total sales = 0 during forecast evaluation region) will not get any recommendation.

Inputs to AutoSource Binary

AutoSource is envoked from a script or the command line. The binary inputs are detailed in the table below:

Binary inputs	Description		Example
-d	Relative or absolute path to domain		-d C:\domains RDF
-mode	Includes the following options:		
	RESTART	Resetting measures, such that the next run starts without prior information. This option does not actually kick off any source level optimization run. Use this option when a clean run is desired, and then run AutoSource with one of the following next modes.	-mode RESTART
	CYCLE	If AutoSource doesn't complete an optimization run due to the time limit, the next time it is run it picks up where it left last time. For instance, if there are 10 source levels and during one run AutoSource only evaluated 3 source levels, then the next time it runs it optimizes source levels 4 and up. CYCLE without a time limit will never finish. Once the last source level was evaluated, AutoSource starts with the first level again.	-mode CYCLE
	ONCEONLY	AutoSource completes the run or stops when the time limit is up	-mode ONCEONLY
-flvllist	A list of the final forecast levels to be optimized		-flvllist 1
Optional binary inputs	**Description**		**Example**
[-today]	Specifies the date when AutoSource stops the evaluation of the forecast error. The evaluation starts at the date given by today minus the number of periods specified in the forecast length. Hence the time interval over which AutoSource evaluates the forecast error is: [today – forecast length, today] The date should be in the RPAS format stored in the dim_day array.		-today D20010101
[-timelimit]	Time, in minutes allowed AutoSource to run. If not specified, there is no time limit.		-timelimit 10,000
[-preserveTemp]	This is a flag indicating if temporary information should be deleted. If not specified, the temporary information is deleted.		-preserveTemp

Example 1:

```
Autosource –mode RESTART –flvllist 1,6 -today DAY20050101 –timelimit 10
Autosource –mode ONCEONLY –flvllist 1,6 -today DAY20050101 –timelimit 10
```

Example 2:

```
Autosource –mode CYCLE –flvllist 1,6 -today DAY20050101 –timelimit 10
```

If only running AutoSource periodically, then use the 'RESTART' and 'ONCEONLY' modes. If the run exceeds the time limit, then 'ONCEONLY' should be run again to pick up where it left off. If you want to start from the beginning again, 'RESTART' and 'ONCEONLY' should be run.

If AutoSource is scheduled as part of the daily cron job, use CYCLE. CYCLE runs RESTART and then ONCEONLY consecutively.

Refer to the *Oracle Retail Demand Forecasting User Guide* for specifics pertaining to the Forecast Maintenance Workbook and picking optimal levels.

Note: For item/stores that are new or highly seasonal, AutoSource may not return the best recommendation since new items may not have an adequate sales history length and highly seasonal items may only sell for a short period during the year. For these items, you should not set the AutoSource recommendation as default at the final level. Only use AutoSource recommendations for item/stores that have an adequate sales history.

AutoSource Measures

The following AutoSource measures are available in the Forecast Maintenance workbook.

Optimal Source Levels

Displayed only at final levels, a value will be populated in this field if AutoSource has been run on the final level. The AutoSource executable evaluates all levels associated to a final level and returns the Source Level that yields the optimal forecast results or lowest error.

Pick Optimal Level

Set only at final levels, a check mark in this field indicates that the batch forecast should use the “Optimal Source Level” selected by “AutoSource.”

The final level measure “Optimal Source Levels” is used for reference. The RDF user can view the “optimal” Source Level that was determined by AutoSource. This Source Level was chosen by generating forecasts at all Source Levels and determining the lowest forecast error (PAE) at the final level.

If the user would like to use the “Optimal Source Level” during forecast generation they can set the “Pick Optimal Level” Boolean measure to TRUE.

If “Pick Optimal Level” is set to TRUE, when forecast generation is run, the “optimal” Source Level will be used. The Forecast Method set at the “optimal” Source Level and the additional associated forecast parameters will also be used.

Usage

Usage: `autosource -d pathToDomain -mode RESTART/ONCEONLY/CYCLE -flvllist lvlx,lvly`

`[-today] todayString(the same format as YYYMMDD)`

`[-timelimit] minutes [-preserveTemp]`

To get this usage text, use `-?`, `-help`, or `-usage`.

To get the version of this utility, use `–version`.

To set the logger verbosity level, use `-loglevel` with values of: `all`, `profile`, `debug`, `information`, `warning`, `error`, or `none`.

To disable the timestamp header, use `–noheader`.

The mode input must be one of RESTART, CYCLE, or ONCEONLY.

The `flvllist` must be a comma separated list of final levels.

The `today` input must be the same format as `YYYMMDD`.

The `timelimit` is in minutes.

- **RESTART**: This mode initializes the system in preparation for a new Autosource batch process.
- **ONCEONLY**: This mode will run the Autosource batch process until it completes or until the timelimit has been reached (whichever comes first).

> **Note:** In order to run in ONCEONLY mode, RESTART mode has to be run first.

- **CYCLE**: This mode will continuously run the Autosource batch process by first running the RESTART mode, and then running ONCEONLY. The CYCLE mode allows the Autosource batch process to always use the latest data in determining the optimal source level for a prod/loc.

Example 1:

```
Autosource –mode RESTART –flvllist 1,6 -today DAY20050101 –timelimit 10
Autosource –mode ONCEONLY –flvllist 1,6 -today DAY20050101 –timelimit 10
```

Example 2:

```
Autosource –mode CYCLE –flvllist 1,6 -today DAY20050101 –timelimit 10
```

If AutoSource is to only run periodically, the 'RESTART' and 'ONCEONLY' modes should be used. If the run exceeds the time limit, then 'ONCEONLY' should be run again to pick up where it left off. If you want to start from the beginning again, 'RESTART' and 'ONCEONLY' should be run.

If AutoSource is to be scheduled as part of the daily cron job, CYCLE should be used. CYCLE will run RESTART then ONCEONLY consecutively.

3

Forecast Approval Alerts

Alerts

Alerts can be configured through the RPAS Configuration Tools or can be manually registered in the domain. The alert expressions require familiarity with the RPAS rule functions. Registering an alert with the alert category of "FORECAST_APPROVAL" allows RDF to use the alert expression during the batch forecasting process to determine if a time series is automatically approved. When this category of alert is registered, the pick lists for Default Approval Method (in Forecast Administration) and the Approval Method Override (in Forecast Maintenance) are updated to include the label of the alert. The user then has the ability to select the alert for any product/location.

The following is an example of Forecast Approval Alert configuration using the example domain that is provided in the release package.

Step 0: Build Global Domain

Using the Mock Install Configuration, build the Global Domain environment.

Step 1 (option 1): Run PreGenerateForecast or Generate

If using a pristine Global Domain or Simple Domain environment, token measures have yet to be registered in the domain(s). Since you do not know the specific birth date at configuration time, token measures allow for measures with birth dates (a time stamp applied during the batch) to be evaluated. The token measure that we are using in this example is System Forecast for level 1 (sf01). The registration of the token measures can be accomplished by running PreGenerateForecast (in a Global Domain environment) or Generate (in a Simple Domain environment). This removes the need to manually execute "regTokenMeasure."

Step 1 (option 2): Use regTokenMeasure to Manually Register Any Token Measures Needed to Support the Alert Expression

If you prefer to manually register the token measures, the regTokenMeasure must be run with –FNHBI option if in a Global Domain environment. This allows the token measures to have different values across subdomains. The token measure requires a value to the measure while registering. In the following example, the token measure is registered in the **Master Domain** and are made to be equal to 'pos' (Weekly Sales) since 'pos' has the same base intersection (item/store/week) and data type (real) as the System Forecast for level 1.

Example:

```
C:\Domains\RDF>regTokenMeasure -d . -add sf01=pos -fnhbi
```

> **Note:** DO NOT do this step if the batch has already been generated since the batch will have automatically registered sf01.

Step 2: Register the Alert Measure

The next step in the process is to register the alert measure **in the Master Domain**. In the following example, an alert with the name of `rdf_a1_1` with label of `Alert1level1` is being registered.

Example:

```
C:\Domains\RDF>regmeasure -d $DOMAIN_DEST_DIR -add "rdf_a1_1" -label
"Alert1level1" -baseint "itemstr_" -db "data/myalerts" -type boolean -
navalue False
```

Step 3: Register the Expression for the Forecast Approval Alert

The `alertmgr` utility is used to register the alert and the alert expression. In the following example, the alert expression first filters out time series with low volume sales (items with forecasts less than three units). This alert compares the System Forecast in the first three weeks in the forecast horizon with last approved forecast for the same three weeks. If the values are within a 33% range, the full forecast horizon is set to automatic approval, otherwise the Alert is triggered. This is all done in batch, so the Alert Manager is NOT necessary to apply the alert. For intersections that do not qualify for automatic approval, the Approval Comment on the Approval Worksheet in the Forecast Approval workbook will contain "refused by alert." You may use the Alert Manager to insert this alert into the workbook to display the intersections that have the alert flag set to true.

Example:

```
C:\Domains\RDF> alertmgr -d . -register "rdf_a1_1" -category
"FORECAST_APPROVAL" -categoryLabel "Alert1level1" -expression
"rdf_a1_1=if(tssum(@sf01,index([clnd].[week],flookup(lfsXLXB.level([data]
.[flvl]+[prod].top),[data].[flvl].[flvl01])),
index([clnd].[week],flookup(lfsXLXB.level([data].[flvl]+[prod].top),[data
].[flvl].[flvl01]))+3)>=3.0,  abs(1-
tssum(@sf01,index([clnd].[week],flookup(lfsXLXB.level([data].[flvl]+[prod
].top),[data].[flvl].[flvl01])),
index([clnd].[week],flookup(lfsXLXB.level([data].[flvl]+[prod].top),[data
].[flvl].[flvl01]))+3)/(tssum(lappf01XB,index([clnd].[week],flookup(lfsXL
XB.level([data].[flvl]+[prod].top),[data].[flvl].[flvl01]))+3,index([clnd
].[week],flookup(lfsXLXB.level([data].[flvl]+[prod].top),[data].[flvl].[f
lvl01])))+0.001))>.33,false)"
```

See the *RPAS Administration Guide* for more information on registering alerts and token measures.

4

RDF Batch Scripts

About RDF Batch Scripts

Batch scripts, which are configured externally, are required by RDF to execute processing before, during and after forecasting. This section outlines a sequence of common batch scripts that may need to be configured during RDF implementation to achieve desired processing to meet your business needs.

1. Hierarchy Loads

   Use the RPAS utility `loadHier` to load standard hierarchy files into RDF. The PRAS utilities `positionBufferMgr` and `reconfigGlobalDomainPartitions` may also need to be called by the `loadHier` script. Refer to the *RPAS Administration Guide* for details on these utilities.

   Please note that RPAS supports centralized hierarchy loads, which means that this script can be executed from the master domain, in a global domain environment.

2. Measure Loads

   Use the RPAS utility `loadMeasure` to load sales history, out of stock information, promotion variables and other necessary data into RDF.

   Please note that RPAS now supports centralized measure loads, which means that this script can be executed from the master domain, in a global domain environment.

3. Preprocessing

   Performs any preprocessing calculations needed for forecasting. This script would invoke the preprocessing rule group that would be configured in the RDF configuration. Preprocessing performs any necessary scrubbing of historic data before forecast generation.

   This script will need to be executed from local domains, if the preprocessing calculations involve RHS and LHS measures that are non-HBI, as would typically be the case for preprocessing calculations. Parallel execution along local domains is possible.

4. Forecast Generation

   In order to generate forecasts, execute the following RDF files:

   - `PreGenerateForecast` – This file is executed from a master domain and performed before running `Generate`.
   - `Generate` – This file is run from local domains in a global domain environment.

5. UpdateFnhbiRdf

   This is an optional script, which is needed only if an alternate hierarchy dimension from the Product hierarchy is used as a dimension in a forecast level.

   This script needs to be run from the master domain, unless it is known that only ONE local domain has forecast data. Then calling this script from that local domain can save some time. Please note that if more than one local domain may have forecast data, then this script needs to be called from the master domain.

6. Alert Manager

   Use the RPAS utility `alertmgr` to evaluate alert conditions specified in the RDF configuration.

With RPAS 12.0.6 and beyond, it is now possible to run `alertmgr` from local domains, followed by a final synchronizing run from the master domain, to synchronize alert hit counts at the master domain level. This can be achieved by running `alertmgr -findAlerts` from the local domains. After running `alertmgr -findAlerts` from the local domains, run `alertmgr-sumAlerts` from the master domain. The `alertmgr -findAlerts` process is more calculation intensive, but it can be performed in parallel at the local domain level.

7. Export Forecasts

   Use the RPAS utility `exportData` to export RDF forecasts from RDF for use by external systems. Users typically export the Approved forecasts from RDF. Also, please refer to the RDF integration scripts (rdf_e_rms.sh) packaged with RPAS.

   This script is run from local domains.

8. Autoworkbook Build

   This script performs any necessary automated workbook builds as set up by the user. Automated workbook builds are set up by users to automate the workbook build process, so that they do not have to make the same wizard selections each time the workbook is built, and they do not have to wait for workbooks to build. Underlying RPAS utility used is `wbbatch`.

   This needs to be run from the local domains.

5

Adding New Local Domains

This section provides an overview on adding new local domains to an existing RDF global domain. New local domains can be added using the RPAS `reconfigGlobalDomainPartitions` utility. Please refer to the *RPAS Administration Guide* for details. It is important to keep in mind that as new local domains are added, they must be added such that the RDF partitioning requirements continue to be met. This means each new local domain can only contain one position along the partition dimension.

When new local domains are added, the following additional scripts will need to be executed, which are located in the /bin directory of $RPAS_HOME:

loadCurveParameters.ksh

This script is used to load the Curve data parameter measures including "Profile Data Source", "Default Source Profile", "Default Profile Approval Method", "Training Window Method", and "Normal Value." This action is typically performed within the plug-ins at domain creation time, however, when you add a new local domain to an existing domain environment, the plug-ins are not run, and therefore this script performs that action manually.

Usage:

```
loadCurveParameters -d fullPathToDomain -s fullPathToNewSubdomain
```

loadRDFParameters.ksh

This script is used to load the RDF data parameter measures including "Default Required Method", "Default Source Level", "Data Plan", "Seasonal Profile" and "Spreading Profile". This action is typically performed within the plugins at domain creation time, however, when you add a new local domain to an existing domain environment, the plugins are not run, and therefore this script performs that action manually.

Usage

```
loadRdfParameters -d fullPathToDomain -s fullPathToNewSubdomain
```

6

Integration

This chapter provides an overview of integration between RDF and the following:

- Integrated Inventory Management Suite
- PPO interface
- RPAS and RDF Integration with RMS

Integrated Inventory Management Suite Interface

The Integrated Inventory Management Suite is the integration of Demand Forecasting (RDF), Advanced Inventory Planning (AIP), Replenishment Optimization (RO), and Analytic Parameter Calculator Replenishment Optimization (APC RO) as a full-suite inventory management solution for retailers.

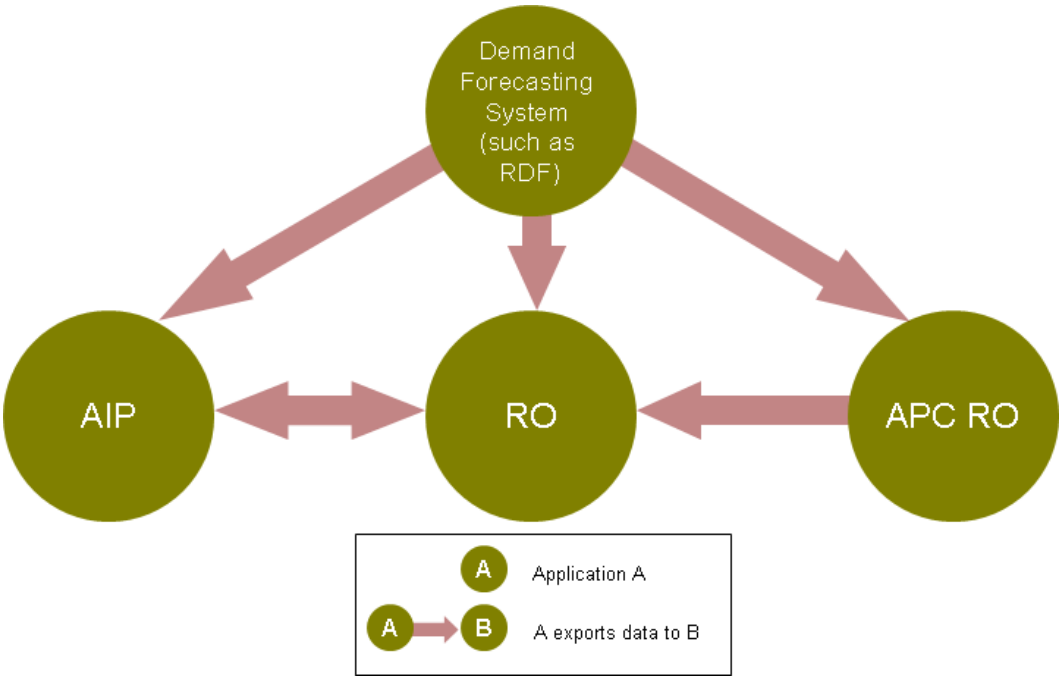


Figure 1: Conceptual Overview

> **Note:** The integration interface between RDF and AIP is not fully certified.

This solution supports data sharing among these applications. Note that the data sharing functionality is not dependent on the presence of all these applications. The defined data sharing between any of the applications works for the entire suite as well as for a subset of the applications.

Integrated Inventory Management Suite Data Flow

Figure 2 shows the integration of the Integrated Inventory Management Suite applications and the flow of data among those applications. Note that the figure below shows a replenishment system. This can be AIP or any other replenishment system. The demand forecasting application can be RDF or any other forecasting system.

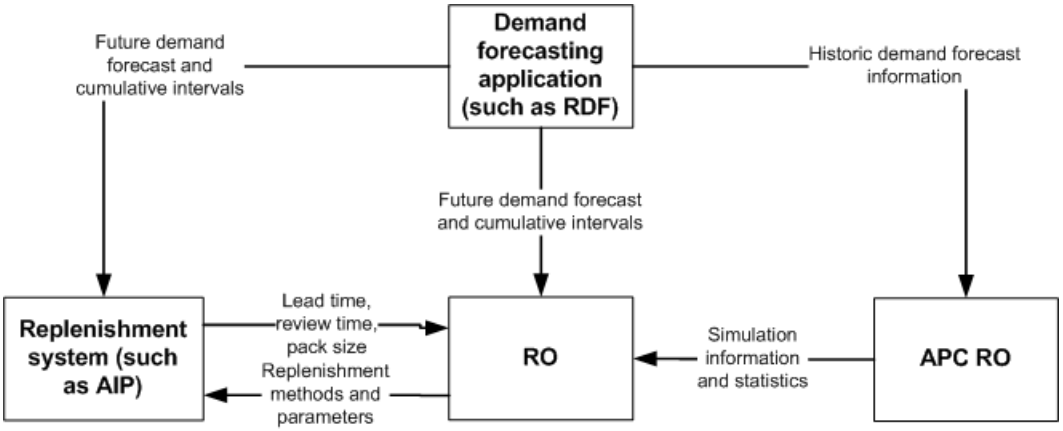


Figure 2: Integrated Inventory Management Suite Data Flow

> **Note:** The integration interface between RDF and AIP is not fully certified.

Data Flow Description

These descriptions explain each of the data flows in Figure 2.

From a Demand Forecasting Application (such as RDF) to APC RO

- Sends historic demand forecasts for a forecast horizon for a series of forecast start dates. It sends a separate forecast file for each forecast start date.

From a Demand Forecasting Application (such as RDF) to a Replenishment System (such as AIP)

- Sends time-phased demand forecasts (starting today and looking forward) at the item/store level.
- Sends the cumulative standard deviation of the forecast. This is needed for the calculation of safety stock.

From APC RO to RO

- Sends simulation information and statistics:
  - Item/location/scenario level information
  - Mean/variability/lead time grouping level information
  - Mean/variability grouping/scenario level information
  - Scenario level information

From a Replenishment System (such as AIP) to RO

- Sends the lead time in flat files. The lead time (or order cycle) pattern generally contains the same lead time on all days that have a lead time; however, the lead time

may increase for the weekend. Therefore the most common lead time is found during the business week.

- Sends the review time in flat files. Review time is the number of days until the next possible receipt. It is a key factor in determining the minimum amount of projected stock that should be available until the next receipt. Since review time can change daily, the minimum available inventory must cover the longest review time in order to avoid stock outs.
- Sends the ordering pack size in flat files. The ordering pack size is the preferred pack size of an item that should be ordered from a source to the destination.

From a Demand Forecasting Application (such as RDF) to RO

- Sends time-phased demand forecasts (starting today and looking forward) at the item/store level. This allows the user to understand how the replenishment settings would perform based on that demand.
- Sends the cumulative standard deviation of the forecast. This is needed for the calculation of safety stock.

From RO to a Replenishment System (such as AIP)

- Sends the recommended replenishment methods and parameters in flat files based on the schedule that the user sets.
- For AIP specifically, RO performs the necessary transformations needed to convert order-based replenishment parameters to a relevant form before sending it to AIP since AIP is a receipts-based system.

Regular Price Optimization Interface

RDF integrates with Regular Price Optimization (RPO) by sending forecasted base demand data, which includes level, trend, and seasonality of estimated demand. It does not, however, include price effects. The forecasted demand data needs to be time-phased, at item/price zone level.

RDF generates the base demand at the item/store level, and RPO aggregates the demand data from the item/store to the item/price zone level.

Figure 3 shows the transfer of data between RDF, and RPO.

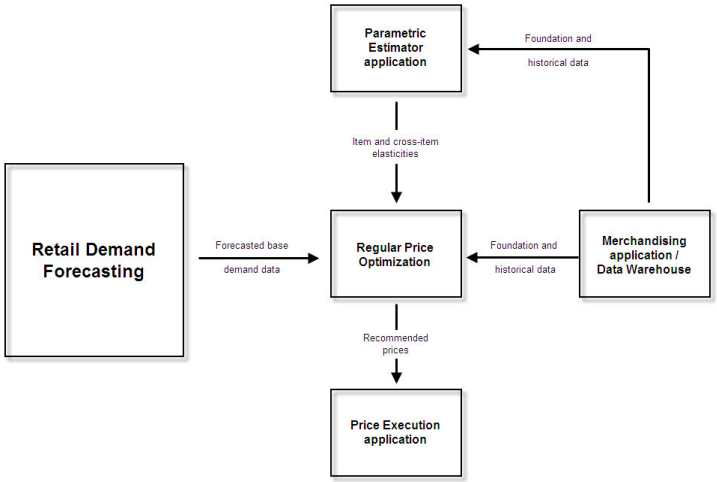


Figure 3: RPO and RDF integration

Promotion Planning and Optimization Interface

RDF integrates with Promotion Planning and Optimization (PPO) by transmitting the future base demand. RDF calculates future base demand using:

- Weekly demand
- Daily weights
- Confidence intervals
- Cumulatve intervals

PPO uses the base demand to plan promotions and determine their financial impact. Figure 4 shows the production environment and the transfer of data between RMS, RDF, and PPO.

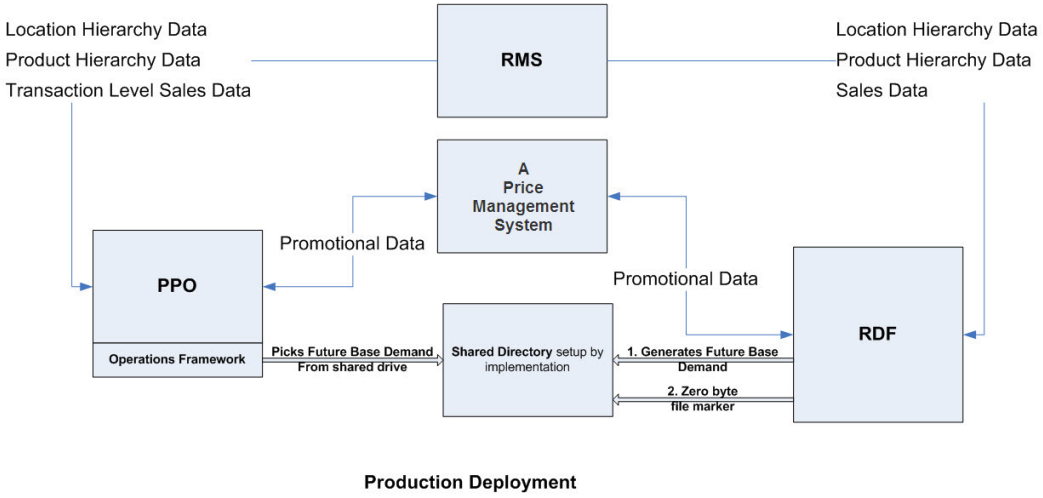


Figure 4: PPO and RDF integration

RPAS and RDF Integration with Retail Merchandising System (RMS)

Summary of Integration Approach with RMS

The strategy for the extraction of foundation data from RMS is for the extract programs (RMSE) to provide flat files in a generic format. For each solution that will use this data, transformation scripts are used to reformat the data as needed to produce a file suitable for loading into the application. For the instances of data coming from RPAS to ha

non-RPAS applications, extract programs are specific to the application in need of the data. Other scripting languages are then used (Perl or AWK) to perform additional data formatting.

This section summarizes the following:

- RMS to RDF transformation programs
  - Merchandise Hierarchy
  - Organization Hierarchy
  - Calendar Hierarchy
  - Store Close Dates
  - Store Open Dates

  - Daily Sales and Issues
  - Weekly Sales and Issues
  - Out of Stock Indicator
- RDF to RMS extract programs
  - Approved Forecasts and Standard Deviations (Cumulative Intervals)
- Grade (RPAS) to RMS extract programs
  - Store Grades
- Curve (RPAS) to Allocation extract programs
  - Differentiator Profiles

Specifics on the usage of RMS extract programs (RMSE's) within the RDF transformation programs are beyond the scope of this document. See the *Oracle Retail Merchandising System Operations Guide* for more information on the RMS extract programs.

Note: For integration compatibility information, see the *RPAS Installation Guide.*

Environment Variable Setup

In addition to any variables identified in the RMS integration documentation, the transformation and/or extract programs require the following environment variables:

- $RPAS_INTEGRATION_HOME: Identifies the location of the integration scripts when /common/header.ksh is run. This variable is used for all integration scripts packaged with the ARPOPlatform EXCEPT those included in 'rfx' (see $RDF_HOME below).
- $TO_RPAS: The staging area for the data to be loaded into RPAS. This directory should be located at the same level as the root of the RPAS domain. For example, if the domain RDF is located in Domains directory (example: /Domains/RDF), then $TO_RPAS should be located at the same level as RDF (example: /Domains/to_rpas).
- $FROM_RPAS: The staging area for the data extract out of RPAS. This directory should be located at the same level as the root of the RPAS domain. For example, if the domain RDF is located in Domains directory (example: /Domains/RDF), then $FROM_RPAS should be located at the same level as RDF (example: /Domains/from_rpas).
- $RDF_HOME: Identifies the location of the root of the 'rfx' directory. The 'rfx' directory packaged with the ARPOPlatform should be added to the location 'rfx' directory packaged with the RMS RETL programs.
- $RI_RMSVERSION: Identifies the version of RMS. If this variable is not set, the integration scripts assume an RMS version of 13. Set the value of this environment variable to 13.

RDF Transformation Programs

Common Program for All Transformations

The **rdft.ksh** script runs all of the necessary data extraction and transformation scripts (**rmse_*.ksh** and **rdft_*.ksh**, respectively) that are needed to produce the files to be loaded into RPAS/RDF/Planning. Most of these scripts are run in parallel (as background jobs).

Usage

```
rdft.ksh  [-x]  [-c] [-sd startDate] [-ed endDate] [-d dir]
```

Arguments:

- `-x`: This option will cause the execution of the RMS data extraction wrapper (rmse.ksh) to be skipped.
- `-c`: This option will cause FILE_DATE in rmse_config.env to be set to the current date instead of using VDATE.
- `-sd`: This option sets the start date for optionally filtering out records based on date. Records with dates prior to this date will be excluded from loading into RDF. The date needs to be in the format YYYYMMDD.
- `-ed`: This option sets the end date for optionally filtering out records based on date. Records with dates after this date will be excluded from loading into RDF. The date needs to be in the format YYYYMMDD.
- `-d`: This option will cause all programs executed by rdft.ksh to be obtained from the "dir" directory.

Transformations of Merchandise Hierarchy Data

rdft_merchhier.ksh is the primary script used to build the data for RPAS from the RMS Merchandise Hierarchy tables. The schema used to produce the output file depends on the attributes and differentiator settings in RMS:

- Case 1: If PROD_ATTRIBUTES_ACTIVE = False and DIFFS_ACTIVE = False, then rdft_merchhier.base.schema will be used to produce the file. In this case, attributes and diff fields will not be included in the merchandise hierarchy file.
- Case 2: If PROD_ATTRIBUTES_ACTIVE = True and DIFFS_ACTIVE = False, rdft_merchhier.attributes.schema will be used to produce the file. The user may choose to have the user-defined attribute fields follow the same concatenation rules as the rest of the columns by setting the variable concat_udas = 1 in the script rdft_merchhier_split_by_domain.ksh. This schema must be manually edited to support a specific attribute model and must be kept in sync with rmse_attributes.schema, rmse_attributes.ksh, and rdft_merchhier_split_by_domain.ksh (see the RMSE end user documentation).
- Case 3: If PROD_ATTRIBUTES_ACTIVE = False and DIFFS_ACTIVE = True, then rdft_merchhier.schema will be used to produce the file. In this case, diff fields will be included in the merchandise hierarchy file.
- Case 4: If PROD_ATTRIBUTES_ACTIVE = True and DIFFS_ACTIVE = True, then an error will result. In this release, the combination of diffs and attributes is not supported.

Intermediate schema and scripts which may be used (depending on configuration options) to produce the merchandise hierarchy file:

- **rdft_diff.domain.schema**
- **rdft_merchdiff.domain.schema**
- **rdft_merchhier_diff_trans.ksh**
- **rdft_merchhier_split_by_domain.ksh**
- **rdft_clean_partition.ksh**

Additional merchandise hierarchy support for issue domains is provided in **rdft_item_loc.ksh**. This script is designed to produce a full item list for issues domains, only containing items that exist in the warehouses.

Note: Issues-specific data transformation functionality is triggered based on the issues setting in RMS (ISSUES_ACTIVE must be set to True).

Transformations of Location Hierarchy Data

rdft_orghier.ksh is the primary script used to build the location data file needed for RPAS from the RMS Organizational Hierarchy Table.

The following five constants may be modified in the script based on location hierarchy data requirements:

- **COMPANY_NAME** – The label for the company position to be populated in the file.
- **COMPANY_ID** – The name for the company position to be populated in the file.
- **STORE_CLASS_CONCAT** – When set to "True", causes the STORE_CLASS to be concatenated on the left of the STORE_CLASS_DESCRIPTION field in the final Store data output file.
- **ADD_AT_SIGN_TO_WH_DESC** – When set to "True", will cause the WHSE_NAME field in the Warehouse ouput file to have an "@" prefix.
- **LONG_WAREHOUSE_RECORDS** – When set to "True", the Warehouse output records will consist of 16 fields. If it is "False", the records will contain only four fields, WH, WHSE_NAME, COMPANY, and CO_NAME.

Intermediate schemas which may be used (depending on configuration options) to produce the location hierarchy file:

- rdft_issues.schema
- rdft_issues_long.schema
- rdft_orghier_store.schema

Note: Issues-specific data transformation functionality is triggered based on the issues setting in RMS (ISSUES_ACTIVE must be set to True).

Transformations of Calendar Hierarchy Data

rdft_calhier.ksh transforms the Calendar Hierarchy data extracted from RMS for loading into RPAS.

Configuration inputs to the script include:

- **DATE_PREF** – The path to the file that contains text indicating whether the format of the Date Description field will be mm/dd/yyyy or dd/mm/yyyy. See the *Oracle Retail Merchandising System Operations Guide* for date format options.
- **LAST_DOW** – The path to the file that contains a day of week name or abbreviation indicating which day of the week is considered to be the end of the week for the fiscal calendar being used at this installation.

Transformations of Daily Sales and Issues Data

rdft_daily_sales.ksh produces the daily sales and issues data files based on regular, promotion, clearance, and issues.

The following constant may be modified in the script based on data requirements:

- **DOM_START_COL** – Defines the starting column position of the Domain ID in the RETL output schema. This is needed by **rdft_merchhier_split_by_domain.ksh** to split the files by domain ID. If the OUTPUT_SCHEMA file is modified, the value of DOM_START_COL may also require modification from the default value.

Intermediate schemas which may be used (depending on configuration options) to produce the sales and/or issues data file:

- rdft_daily_sales.schema

> **Note:** Issues-specific data transformation functionality is triggered based on the issues setting in RMS (ISSUES_ACTIVE must be set to True).

Transformations of Weekly Sales and Issues Data

rdft_weekly_sales.ksh produces the weekly sales and issues data files based on regular, promotion, clearance and issues.

The following constant may be modified in the script based on data requirements

- **DOM_START_COL** – Defines the starting column position of the Domain ID in the RETL output schema. This is needed by rdft_merchhier_split_by_domain.ksh to split the files by domain ID. If the OUTPUT_SCHEMA file is modified, the value of DOM_START_COL may also require modification from the default value.

Intermediate schemas which may be used (depending on configuration options) to produce the sales and/or issues data files:

- rdft_weekly_sales.schema

> **Note:** Issues-specific data transformation functionality is triggered based on the issues setting in RMS (ISSUES_ACTIVE must be set to True).

Transformations of Store Open Date Data

rdft_open_date.ksh produces the Store/Warehouse Opening Date data file.

Intermediate schema used to produce the store open date data files:

- rdft_open_date.schema

Transformations of Store Close Date Data

rdft_close_date.ksh produces the Store/Warehouse Closing Date data file.

Intermediate schema used to produce the store closing date data files:

- rdft_close_date.schema

Transformations of Out of Stock Indicator Data

rdft_outofstock.ksh produces the Store and Warehouse (issues) Out of Stock Indicator data extracted from RMS.

Intermediate schema and scripts which may be used (depending on configuration options) to produce the Out of Stock Indicator data file:

- rdft_outstock_split_by_domain.awk
- rdft_outofstock.schema
- rdft_outofstock_issues.schema
- rdft_outofstock_sales.schema

RDF Transformation Matrix

The following matrix identifies the transformation scripts and schemas used for each the hierarchy and data files produced for RDF:

Directory	Script or Schema Name	Merchandise Hierarchy	Location Hierarchy	Calendar	Daily Sales & Issues	Weekly Sales & Issues	Out of Stock Indicator	Store Open Dates	Store Close Dates
rfx/lib	rdft_merchhier_diff_trans.ksh	X							
	rdft_merchhier_split_by_domain.ksh	X							
	rdft_outofstock_split_by_domain.ksh						X		
rfx/schema	rdft_close_date.schema								X
	rdft_daily_sales.schema				X				
	rdft_diff.domain.schema	X							
	rdft_merchierdiff.domain.schema	X							
	rdft_merchier.attributes.schema	X							
	rdft_merchhier.base.schema	X							
	rdft_merchhier.domain.schema	X							
	rdft_merchhier.schema	X							
	rdft_open_date.schema							X	
	rdft_orghier_issues.schema		X						
	rdft_orghier_issues_long.schema		X						
	rdft_orghier_strore.schema		X						
	rdft_outofstock.schema						X		
	rdft_outofstock_issues.schema						X		
	rdft_outofstock_sales.schema						X		
	rdft_weekly_sales.schema					X			
rfx/src	rdft_ksh	X	X	X	X	X	X	X	X
	rdft_calhier.ksh			X					
	rdft_clean_partition.ksh	X							
	rdft_close_date.ksh								X
	rdft_daily_sales.ksh				X				
	rdft_item_loc.ksh	X							
	rdft_merchhier.ksh	X							
	rdft_open_date.ksh							X	
	rdft_orghier.ksh		X						
	rdft_outofstock.ksh						X		
	rdft_weekly_sales.ksh					X			

Common Programs for Extracts

config.ksh is a configuration directory that requires both the RMS version being integrated and the backup action to be defined.

The following OPTIONAL arguments are available:

- Name of the domain: Defaults to directory name
- Number of the domain: Defaults to the 2 last digits of the directory name
- Format of timestamp attached to logs and processed input files: Defaults to: (date +"%b%d%a%I%M%p") (example: Aug02Thu0111PM)
- Data Drop: Defaults to ../../to_rpas
- Data Export: Defaults to ../../from_rpas
- Log Drop: Defaults to ./logs
- Error Drop: Defaults to ./err
- Reclass Data: Defaults to ../reclass_data

functions.ksh

This script file contains ksh functions that are used by scripts in [DOM]/scripts. It should be sourced, not executed in order to preserve environment variables.

header.ksh

This script should be run at the beginning of any implementation-specific script to setup function libraries, environment, and platform-specific routines.

Extract of Forecast Data for RMS

rdf_e_rms.ksh extracts forecast demand value and standard deviation (cumulative interval) at both day and week aggregations from an RDF domain.

Arguments:

- `-t`: <Domain Type> (S for sales, I for issues)
- `-w`: <Data Width> ([7...18], defaults to 12)
- `-d`: <Domain> (defaults to current directory)
- `-n`: <Domain Number> (defaults to last two digits of domain)

Output files:

${RPAS_EXPORT}/d<s|i>demand.<Domain Number> (demand at day)

${RPAS_EXPORT}/w<s|i>demand.<Domain Number> (demand at week)

The following table provides information about the output file data format.

Field	Start	Width	Format
Day | EOW Day	1	8	Alpha
Product ID	1	25	Alpha
Location ID	26	20	Alpha
Demand	46	12	Alpha
Std. Dev. Demand	68*	12*	Numeric (floating point, 4 decimal digits with decimal)

* Width of Demand and Std. Dev. Demand may be overridden with the `-w` parameter; stated values Demand width and Std. Dev. Demand start and width are based on default width of 12.

> **Note:** the following must be defined in the shell environment prior to calling this script:
>
> – RPAS_HOME
>
> – RPAS_INTEGRATION_HOME

Editing for Simple Domains

If you have a simple domain, then the functions.ksh script needs to be edited.

Under the CreateWeek2DayArray () function call, edit this line by removing the `-fnhbi` flag.

Edit from:

```
regmeasure -d $RPAS_DOMAIN -add WEEK2DATE -type date -baseint week -fnhbi
-db data/hmaint
```

to:

```
regmeasure -d $RPAS_DOMAIN -add WEEK2DATE -type date -baseint week -db
data/hmaint
```

Load of Extracted Forecast Data and Standard Deviations to RMS

rmsl_forecast.ksh pulls the daily/weekly forecast items into RMS.

During the loading of each domain file the following steps are performed:

1. Truncate the partition in the RMS forecast table which corresponds to the domain ID.

   > **Note:** Partition names should always be in the format: [tablename]_[domainID]

2. Append a domain field and insert the domain_id into each record.
3. Load the forecast data into the RMS forecast table.

   Example: `rmsl_rpas_forecast.ksh  daily | weekly`

Intermediate schemas which may be used (depending on configuration options) to produce the forecast data files:

- rmsl_forecast_daily.schema
- rmsl_forecast_weekly.schema

Extract of Diff Profile Data for Allocation

profile_e_alloc.ksh extracts Curve diff profiles for use by Allocation.

Arguments:

- `-p`: <Profile Number>
- `-m`: <Mask Measure> (Optional mask; only positions for which the mask value is non-NA will be exported.)
- `-w`: <Data Width> ([7...18], defaults to 12)
- `-d`: <Domain> (defaults to current directory)
- `-n`: <Domain Number> (defaults to last two digits of domain)

Output file: ${RPAS_EXPORT}/d1<Product Level>.<Domain Number>, where Product Level is the Aggregation intersection's Prod dimension

The following table provides information about the output file data format.

Field	Start	Width	Format
Product ID	1	25	Alpha
Location ID	26	20	Alpha
Diff ID (optional)	46	36	Alpha
Quantity	82	12*	Numeric (floating point, 4 decimal digits, no decimal)*
Std. Dev. Demand	68*	12*	Numeric (floating point, 4 decimal digits with decimal)

* Quantity width may be overridden with the `-w` parameter.

Note: The following must be defined in the shell environment prior to calling this script:

– RPAS_HOME

– RPAS_INTEGRATION_HOME

Extract of Store Grade Data for RMS

grade_e_rms.ksh extracts store grades for use by RMS.

Arguments:

- `-t` <Timestamp> (YYMMDDTTTT). This value corresponds to the timestamp of the Cluster Membership measure (clpm+<Timestamp>) to be extracted
- `-d` <Domain> (defaults to current directory)
- `-n` <Domain Number> (defaults to last two digits of domain)

Output file: ${RPAS_EXPORT}/gr<Timestamp>.<Domain Number>

Output file data format:

(a thru c) constitutes the header records

- (d thru j) constitutes the detail records
- (k thu l) constitutes the footer records
  - **a.** 'FHEAD'
  - **b.** Line ID Number
  - **c.** 'GRADU'
  - **d.** 'FDETL' (Record Identifier)
  - **e.** Line Sequence Identifier
  - **f.** Grade Group ID Number: This value corresponds to the first 8 characters of the Cluster Run Name measure (clnam+<user-defined name>) set by the user in the Generate Cluster wizard in Grade. For integration with RMS, the Cluster Run Name must be populated with only numeric characters.
  - **g.** Grade Group: This value corresponds to the first "**N**" characters of the Cluster Run Name measure (clnam+<user-defined name>) set by the user in the Generate Cluster wizard in Grade. "**N**" is 20 for RMS version 11.0, and "**N**" is 120 for RMS version 12. The script determines the RMS version from the environment variable RI_RMSVERSION. RMS version 12 is assigned by default.

h. Store ID
i. Grade Member Name
j. 'FTAIL' (Record Identifier)
k. Line ID Number
l. FDETL Line Total Number

Note: The following must be defined in the shell environment prior to calling this script:

– RPAS_HOME

– RPAS_INTEGRATION_HOME

RDF Extract Matrix

The following matrix identifies the extract scripts and schemas used for each the data files produced for RMS:

Directory	Script or Schema Name	Forecasts and Standard Deviations	Diff Profiles
common	config.ksh		
	functions.ksh	X	
	header.ksh	X	X
curve	profile_e_alloc.ksh		X
grade	grade_e_rms.ksh		
plan	Plan_e_alloc.ksh		
	Plan_e_price.ksh		
	Plan_e_plcblwdm.ksh		
	Plan_e_ploblwdm.ksh		
rdf	rdf_e_rms.ksh	X	
	rmsl_forecast.ksh	X	
	rmsl_forecast_daily.schema	X	
	rmsl_forecast_weekly.schema	X	

7

Internationalization

Internationalization is the process of creating software that is able to be translated more easily. Changes to the code are not specific to any particular market. Oracle Retail Predictive Solutions have been internationalized to support multiple languages.

This chapter describes configuration settings and features of the software that ensure that the base application can handle multiple languages.

Translation

Translation is the process of interpreting and adapting text from one language into another. Although the code itself is not translated, components of the application that are translated include the following:

- Graphical user interface (GUI)
- Error messages

The following components are not translated:

- Documentation (Online Help, Release Notes, Installation Guide, User Guide, Operations Guide)
- Batch programs and messages
- Log files
- Configuration Tools
- Reports
- Demo data
- Training Materials

The user interface for Oracle Retail Predictive Solutions has been translated into:

- Brazilian Portuguese
- French
- German
- Italian
- Japanese
- Korean
- Russian
- Simplified Chinese
- Spanish
- Traditional Chinese

Note: For more information about internationalization, see the *RPAS Administration Guide*.